Product Specification For XEP80
80 Column and Parallel Printer Board

01/26/87

XEP80 Firmware   - Rev 41.1
6502 XE Handler   - Rev 70.0

Table of Contents

1.0 General Description

1.1 Packaging Information

The XEP80 package consists of the following parts:

1. Owners Manual
2. RCA type video cable assembly
3. XEP80 System Diskette
4. Warranty Card
5. XEP80 Unit
6. Two End Caps
7. Power Adapter - 2600 Game Type
8. Box for Power Adapter
9. Gift Box

1.2 Statistics

XEP80 Current Draw 400mA

Power Supply 500mA 9V DC

Physical Dimensions - 40mmH 240mmL 142mmW

Video Output - NTSC Composite Monochrome

Video Display:
Text - 80 columns by 25 lines
Graphics - 320 dots by 200 lines

Parallel Port:
Pin information - 8 data bits, strobe, busy, ground
Connector - DB25 female

1.3 General Information

The XEP80 is a video output device that also supports an 8 bit type parallel port. The video hardware generates an 80 column by 25 line text display through a video connector/cable plugged into a separately purchased monitor. The monitor can be any Composite Video input type, although for the best display a monochrome is strongly advised.

Internally, the XEP80 is a 256 character wide by 25 line high storage device with an 80 column wide display window. Characters may be placed anywhere within the device independent of the window location. The window may be scrolled across the 256 column wide field.

Optionally, the XEP80 may be placed into Pixel Graphics mode. This mode supports a bit mapped (pixel) screen of 320 dots (40 bytes) horizontal by 200 dots (lines) vertical. The output window displayed is approximately half the size of the text window.

Communication to and from the XEP80 is established through a joystick type cable that extends from the back of the unit. This cable will plug into either joystick port (1 or 2) on all Atari 8 bit computers. Through this cable the XEP80 receives commands and data from the computer and sends them to either the screen or the printer.

The 8 bit parallel port allows Epson, Centronics, etc. type printers to be connected to the XEP80, which means the Atari Computer may now output to these printers. The parallel connector is a female 25 pin D type.

Audio may be supplied via a video cable from the Atari computer DIN plug to any suitable audio speaker input. This is required for any SOUND commands or keypresses.

Software is provided on diskette in the form of a DOS binary file named AUTORUN.SYS. This file is automatically loaded into memory and initialized by the DOS at boot (power on) time.

NOTE: The "*" character used throughout this document means that a 0 or 1 may be substituted. In the case of an 8 bit value, the description "0 0 * * * * * *" will allow 00 through $3F as valid input.

2.0 Physical Interface

The Joystick port will use two lines for interfacing. One PIA line for input from the XEP80 and one PIA line for output to XEP80. Both the XEP80 and computer will share a common ground line. Either joystick port (1 or 2) may be used with the supplied software.

MALE (XE/XL/800/400) COMMUNICATION PORT PIN CONFIGURATIONS:

```
          JOYSTICK 1                         JOYSTICK 2

    ______________________      ______________________
    \  1   2   3   4   5 /      \ 10  11  12  13  14 /
     \                  /        \                  /
      \  6   7   8   9 /          \ 15  16  17  18 /
       \______________/            \______________/
```

PIN	FUNCTION
1	Data to XEP80
2	Data from XEP80
3-7	Not Used
8	Common Ground
9	Not Used
10	Data to XEP80
11	Data from XEP80
12-16	Not Used
17	Common Ground
18	Not Used

FEMALE (XEP80) PARALLEL PRINTER PORT PIN CONFIGURATION:

```
    __________________________________________
    \ 13 12 11 10  9  8  7  6  5  4  3  2  1 /
     \  25 24 23 22 21 20 19 18 17 16 15 14 /
      \____________________________________/
```

PIN	FUNCTION
1	Strobe
2-9	Parallel Data
10	Not Used
11	Busy
12-17	Not Used
18-25	Ground

3.0 Software Interface

3.1 General Description

The software interface to the XEP80 on the 8 bit computer side is a disk booted handler. This handler replaces the E:, S:, and P: vectors in the Handler Address Table ($31A).

Loading of the handler is as follows: A DOS disk containing the handler must be in drive 1. The computer is powered on, causing a disk boot. The DOS is loaded in and initialized. The DOS init code looks for the file AUTORUN.SYS on the boot disk and loads it. The XEP80 handler must be named AUTORUN.SYS to be loaded at boot time. The handler file is loaded into RAM at address $3000 and continuing upward approximately 1.5K bytes.

The file consists of a relocater portion followed by the actual XEP80 handler. After the file is in memory, DOS causes the init code of the handler to execute. This init code is the relocater which uses a variable called MEMTOP ($2E5,2E6) to find the last address of user (non-system) RAM. MEMTOP is then decremented by the relocater to point to an address about 1.5K below its old value. This will make room for the handler portion to be moved to those addresses. This address space into which the handler will be loaded is just below the normal screen RAM and display list address space.

While the relocater moves the handler to its proper location, it resolves all absolute address references within the handler code. Thus the handler code comes from disk in a relocating format (see Appendix B). After the handler has been moved, the relocater sets the DOSINI vector ($0C) to point to the handler init code, sets an address within the handler to point to the old DOSINI code, then jumps to the handler init code.

The handler init sequence sets up the PIA registers to communicate to the XEP80, modifies the Handler Address Table (HATABS $31A), sets new screen margins, and determines whether the system is NTSC (60 Hz) or PAL (50 Hz). At this point the init code checks the position of the SHIFT key. If the key is pressed the code leaves the 40 column screen on and exits. If the key is not pressed the code then disables the ANTIC chip from display so that XEP80 I/O can begin.

The handler now exits back to DOS, which then exits back to the OS, which then decides which application to run (for example the BASIC cartridge) and jumps to that code. As long as the application makes I/O calls for the screen and printer through the normal OS CIO calling conventions, the XEP80 handler will intercept them and pass data on to the XEP80 device.

3.2 Functional Description

3.2.1 Screen Operations

The handler will monitor and communicate the following locations to the XEP80 and will update them if the XEP80 causes a change:

$055	Cursor X Position COLCRS (low byte)
$056	Cursor X Position (high) for range checking
$054	Cursor Y Position ROWCRS

The following locations will be monitored and communicated to the XEP80:

		Default:	Optional:
$2FE	List Flag DSPFLG	$00 (OFF)	$01 (ON)
$2F0	Cursor On/Off Flag CRSINH ...	$00 (ON)	$01 (OFF)
$052	Cursor Left Margin LMARGN ...	$00 (XEP80)	$XX (ANY)
$053	Cursor Right Margin RMARGN ..	$4F (XEP80)	$XX (ANY)
$2F4	Character Base CHBAS	$E0 (ATASCII)	$CC (ICM)

The following locations will be monitored and acted upon by the handler to emulate screen editor operations:

$2FF	Start/Stop Screen Output Flag
Y Reg	EOF Status (CNTL 3 Keystroke) and BREAK Key

Logical lines of text on the screen may encompass more than one physical line. The length of each physical line within a logical line is determined by the left and right margins. Up to 24 physical lines may be included within one logical line.

Tabs are set at 2,7,15,23,31,39,47,55,63,71,79,87,95,103,111,119, 127,135,143,151,159,167,175,183,191,199,207,215,223,231,239,247,255. These are the default settings and may be changed with the set and clear tab control characters. Tab values only apply to physical lines, thus a logical line of two or more physical lines will have its tab stops repeated on each physical line.

The 40 column ANTIC screen may be enabled with a command in the event the user wishes color graphics, 40 column text, or has no 80 column monitor. The parallel printer is still operational in this mode.

Graphics is enabled by a special OPEN command to "S:". In this mode the screen is 320 dots (40 bytes) by 200 lines. The cursor will be in the upper left hand corner of the screen awaiting data transfer. Each character sent to the XEP80 will be displayed in sequence; all 8 bits will correspond to consecutive dots across a scan line. The MSB will be displayed first and the LSB is last.

The status line is Y cursor value 24. This line may be used for input or output, however control characters do not edit, but only print in this line. One control is allowed, SHIFT DELETE. The line does not scroll. Be careful not to accidently POSITION or POKE the cursor into this line while the BASIC command line processor is active as it may require use of the Reset key to recover from a lockup that can occur on this line.

3.2.1.1 Editor Screen "E:" Handler

The following functions will be supported by the supplied handler and will emulate the 8 bit screen handler (editor):

Open — Assign edit handler to IOCB #N and clear screen. If AUX1 is $2C screen will not be cleared.

Close — Unassign handler from IOCB #N.

Get Char — Get line of text until EOL (INPUT in Basic). The special read mode (read a line of the screen without keyboard input) is also supported by setting AUX1 of calling IOCB to $0D.

Put Char — Put a character to the screen or do editing functions (PRINT or PUT in Basic). See chart below.

Get Status — Return default "on line" status

Special — Send commands to and receive status from XEP80. (XIO in Basic). AUX1 byte must be $0C, AUX2 is code to send, CMD below.

Put Character Values

ATASCII	FUNCTION
$1B	Escape - Print next char even if a control char
$1C	Cursor Up
$1D	Cursor Down
$1E	Cursor Left
$1F	Cursor Right
$7D	Clear Screen
$7E	Backspace
$7F	Tab
$9B	Atari EOL (CR/LF)
$9C	Delete Line
$9D	Insert Line
$9E	Clear Tab
$9F	Set Tab
$FD	Sound Bell - Not Available (See item 6 below)
$FE	Delete Character
$FF	Insert Character

Items:
1) Put Char will advance cursor.
2) Get Char will advance cursor.
3) At logical end of line a new line will be inserted.
4) At non EOL put will overwrite characters.
5) The XEP80 E: handler may be disabled by pressing the SHIFT key at power on or during RESET.
6) The bell sound has been disabled by the handler to improve the speed of character output to the XEP80. An application may, however, use any of the SOUND statements (or the assembly language equivalents) to generate noise, notes, etc.

Special Commands to "E:"

CMD	AUX1	AUX2	Function
$14	$0C	CMD	- send command ($00-BF,D0-DF) to XEP80 (see 4.3.2)
$15	$0C	00	- set normal character transmit mode
$15	$0C	01	- set burst character transmit mode
$16	$0C	CMD	- send input command ($C1-C6) to XEP80 (see 4.3.4) response is stored in location $2EB
$18	$0C	00	- exit 40 col screen, enter and clear XEP80 screen
$18	$2C	00	- exit 40 col screen, enter 80 with no clear screen
$19	$0C	00	- exit XEP80 screen, enter 40 col screen

3.2.1.2 Static Screen "S:" Handler

Open	Assign screen handler to IOCB #N (Graphics see below)
Close	Unassign handler from IOCB #N
Get Char	Get a character and advance cursor (Basic LOCATE)
Put Char	Put a character to the screen - no editing functions
Get Status	Return default "on line" status
Special	No action

Items:
1) Put char will advance cursor and possibly insert lines.
2) Get char will advance cursor but not insert lines.
3) Get char over blank will return $20 (SPACE).
4) Put EOL ($9B) will move cursor to left margin of next line.
5) A GRAPHICS 8+16 will set graphics mode on the XEP80. Or if IOCB #N is opened with AUX1=$1C and AUX2=08.
6) When in graphics mode on the XEP80 the cursor is initially in the upper left hand corner of the graphics window. Each byte sent to the XEP80 will be placed on the screen as a series of 8 dots (MSB to LSB) corresponding to the binary value of the byte or ATASCII value of the character. The cursor then moves over 8 dots to the right to show where the next byte will be placed. When the cursor reaches the right edge of the window it moves to the left edge and down one line.
7) Currently Put char sends ESC followed by the character. This will give two bytes per put in graphics mode. Therefore use Put char through E: when transmitting graphics data.
8) The graphics mode command automatically sets burst mode.
9) The handler will support standard ANTIC graphics calls while in 40 col mode, provided enough room has been allocated in RAM. For normal boot environments this means GRAPHICS 1,2, 3,4 only.

3.2.2 Printer Operations

The XEP80 will also output single characters to the printer port through a 2K internal buffer. Data presented to the printer by the handler may be interpreted in one of three ways:

No Translation - All 8 data bits of character information will be passed to the printer port.

Light Translation - ATASCII EOL ($9B) is translated by the printer handler to the CR and LF characters before transmission to the printer. This is the default printer mode. Optionally, only the CR may be sent for EOL.

Heavy Translation - Light translation plus ASCII characters below $20 will be translated to a "^" character (for CONTROL) followed by the corresponding letter before being sent to the printer. The MSB will be set to 0.

The printer output may be directed to either the XEP80 (parallel) or serial ports or both. The unit number (such as P2:) may be kept or altered depending upon the need. See chart below.

Calls to "P:"

Open	Assign printer handler to IOCB #N and issue status request to printer. AUX1 must be 08 (Write only).
Close	unassign handler from IOCB #N
Get Char	Not supported
Put Char	Send a character to the printer port
Get Status	Returns status of printer BUSY line, 1=available, 139=busy or off line.
Special	Configure printers and select translate mode (see chart below). Do before OPEN to printer.

Special Configuration Command to P: Command byte =$17

Configuration Byte Assignments for AUX1

```
+-+-+-+-+-+-+-+-+
|7|6|5|4|3|2|1|0|     Bit positions
+-+-+-+-+-+-+-+-+

 0 0 0 0 0 * * 0      P1: - Parallel Port
 0 0 0 0 0 * * 1      P1: - Serial Port as P1:
 0 0 0 0 0 0 0 *      P2: - Serial Port as P2:
 0 0 0 0 0 0 1 *      P2: - Serial Port as P1:
 0 0 0 0 0 1 0 *      Reserved
 0 0 0 0 0 1 1 *      P2: - Parallel Port
 0 0 0 0 1 * * *      Reserved
 0 0 0 1 * * * *      "
 0 0 1 * * * * *      "
 0 1 * * * * * *      "
 1 * * * * * * *      "
```

Configuration Byte Assignments for AUX2

```
 0 0 0 0 0 0 0 0      Light translation - EOL to CR/LF
 0 0 0 0 0 0 0 1      No translation
 0 0 0 0 0 0 1 0      Heavy translation - EOL to CR/LF
 0 0 0 0 0 0 1 1      Reserved
 0 0 0 0 0 1 0 0      Light translation - EOL to CR
 0 0 0 0 0 1 0 1      No translation
 0 0 0 0 0 1 1 0      Heavy translation - EOL to CR
 0 0 0 0 0 1 1 1      Reserved
 0 0 0 0 1 * * *      "
 0 0 0 1 * * * *      "
 0 0 1 * * * * *      "
 0 1 * * * * * *      "
 1 * * * * * * *      "
```

Note - P3 through P15 are not alterable; they remain unaffected by the handler and are directed only over the serial bus.

4.0 Communications Interface

4.1 Communication Fundamentals

Communication with the XEP80 will be handled by sending serialized data over bit 0 or 4 (JOY0,JOY1) and receiving serialized data over bit 1 or 5 (JOY0,JOY1) of the PIA A I/O register. Data will be shifted at 15.7KB. Transmitted and received data will each contain 9 bits of information . Data will be shifted out as follows: a start bit of value zero, followed by bits D0 through D7, followed next by a control bit to indicate whether data (0) or command (1) information has been sent, then terminated with a stop bit of value 1. Currently the supplied handler will provide data shifting on the computer end. Any applications that do not use the handler or drive the XEP80 at a different baud rate must provide code for this. It must be understood that this code is time critical in operation, so all 6502/ANTIC cycles must be accounted for during data shifting.

An alternate transmit/receive mode, "burst mode", allows faster transmit times. The XEP80 to computer data line (receive line) is high (1) when characters may be sent and it is a low (0) when the XEP80 is busy. Thus no cursor information is returned. This mode applies only to characters to be sent to the screen (including control characters) and characters sent to the printer (the only mode for the printer). Burst mode does not apply to commands of any sort.

Any custom handler that sets burst mode must be aware of how to handle communication from that point.

When waiting for a cursor or character (non burst mode) monitor the data-in line immediately after transmitting a byte.

When in burst mode wait at least 90 usec before testing the data-in line for 0 (busy) or 1 (ready).

Do not allow any 6502 interrupts when transmitting or receiving the serialized data, as this will modify the baud rate timings.

Interrupts may be allowed while waiting for the XEP80 to go ready in burst mode. Thus other interrupt driven processes such as a modem handler will not lose characters.

4.2 Communication Protocol

Characters sent to the XEP80 will cause a 1 or 2 byte cursor to be returned - see 4.3.1 and 4.3.6

Commands are subdivided into three groups:

1) Stand alone commands - no response expected - see 4.3.2
2) Get Character - character and cursor returned - see 4.3.3
3) Get Data commands - data byte only returned - see 4.3.4

Cursor values returned from XEP80 are determined by following logic:

1) If only Y cursor has changed, only Y is returned
2) If only X cursor has changed, only X is returned
3) If both have changed both are returned (X is returned first)
4) If neither changes only X is returned

```
COMPUTER    XEP80

1  CMD-->                 Stand alone command


2  CMD-->                 Get data command
         <--CHAR          Return value


3  CMD-->                 Get character command
         <--CHAR          Return character
         <--CURSOR X      and X cursor


4  CMD-->                 Get character command
         <--CHAR          Return character
         <--CURSOR Y      and Y cursor


5  CMD-->                 Get character command
         <--CHAR          Return character
         <--CURSOR X      and X and Y cursor
         <--CURSOR Y


6  CHAR->                 Output character
         <--CURSOR X      Return X cursor


7  CHAR->                 Output character
         <--CURSOR Y      Return Y cursor


8  CHAR->                 Output character
         <--CURSOR X      Return X and Y cursor
         <--CURSOR Y

9  CHAR->                 Burst Mode - send character
       ___.     .___      High (1) Ready
          |_____|         Low (0) Busy
```

4.3 Command/Data Values

4.3.1 Data Output - Return Cursor

```
    +-+-+-+-+-+-+-+-+-+
    |8|7|6|5|4|3|2|1|0|
    +-+-+-+-+-+-+-+-+-+

     0 x x x x x x x x    Character to be output ($00-FF) (3.2.1.1, 3.2.2)
```

4.3.2 Stand Alone Command - No Information Returned

```
    +-+-+-+-+-+-+-+-+-+
    |8|7|6|5|4|3|2|1|0|
    +-+-+-+-+-+-+-+-+-+

     1 0 0 x x x x x x    Horizontal Cursor Position ($00-3F)
     1 0 1 0 0 x x x x    Horizontal Cursor Position ($40-4F)
     1 0 1 0 1 x x x x    Horiz Curs Pos High Nibble - for wide screen
     1 0 1 1 0 x x x x    Left Margin Low Nibble - sets high nibble to 00
     1 0 1 1 1 x x x x    Left Margin High Nibble
     1 1 0 0 0 x x x x    Vertical Cursor Position ($00-0F)
     1 1 0 0 1 0 x x x    Vertical Cursor Position ($10-17)
     1 1 0 0 1 1 0 0 0    Set Cursor to Status Row ($18) See caution, Pg. 6
     1 1 0 0 1 1 0 0 1    Set Graphics to 60 Hz
     1 1 0 0 1 1 0 1 0    Modify Graphics to 50 Hz
     1 1 0 0 1 1 0 1 1    Reserved
     1 1 0 0 1 1 1 x x    Reserved
     1 1 0 1 0 x x x x    Right Margin Low Nibble - sets high nibble to 04
     1 1 0 1 1 x x x x    Right Margin High Nibble
     1 1 1 0 1 0 0 0 0    Clear List Flag
     1 1 1 0 1 0 0 0 1    Set List Flag
     1 1 1 0 1 0 0 1 0    Set Screen Normal Mode - cursor returned each char
     1 1 1 0 1 0 0 1 1    Set Screen Burst Mode - no cursor returned
     1 1 1 0 1 0 1 0 0    Select Character Set A - Atari graphics (ATASCII)
     1 1 1 0 1 0 1 0 1    Select Character Set B - Atari international
     1 1 1 0 1 0 1 1 0    Select XEP80 Internal Character Set
     1 1 1 0 1 0 1 1 1    Modify Text to 50 Hz Operation
     1 1 1 0 1 1 0 0 0    Cursor Off
     1 1 1 0 1 1 0 0 1    Cursor On Continuous
     1 1 1 0 1 1 0 1 0    Cursor On Blink
     1 1 1 0 1 1 0 1 1    Move Cursor to Start of Logical Line
     1 1 1 0 1 1 1 0 0    Set Scroll Window to Cursor X Value
     1 1 1 0 1 1 1 0 1    Set Printer Output
     1 1 1 0 1 1 1 1 0    Select White Characters on Black Background
     1 1 1 0 1 1 1 1 1    Select Black Characters on White Background
     1 1 1 1 x x x x x    Reserved
```

4.3.3 Get Character Command - Data byte and cursor returned (4.3.5, 4.3.6)

```
+-+-+-+-+-+-+-+-+-+
|8|7|6|5|4|3|2|1|0|
+-+-+-+-+-+-+-+-+-+
```

1 1 1 0 0 0 0 0 0	Get Character from XEP80 at cursor (and advance)

4.3.4 Get Data Command - Data byte returned (4.3.5)

```
+-+-+-+-+-+-+-+-+-+
|8|7|6|5|4|3|2|1|0|
+-+-+-+-+-+-+-+-+-+
```

	Command Function for XEP80	Byte Returned
1 1 1 0 0 0 0 0 1	Request Horizontal Cursor	X Cursor Value
1 1 1 0 0 0 0 1 0	Master Reset	01
1 1 1 0 0 0 0 1 1	Printer Port Status	00=busy 01=on line
1 1 1 0 0 0 1 0 0	Fill RAM with previous char	01
1 1 1 0 0 0 1 0 1	Fill RAM with space ($20)	01
1 1 1 0 0 0 1 1 0	Fill RAM with EOL ($9B)	01
1 1 1 0 0 0 1 1 1	Reserved	
1 1 1 0 0 1 * * *	"	

4.3.5 Received Data - Normal Communication Mode

```
    +-+-+-+-+-+-+-+-+-+
    |8|7|6|5|4|3|2|1|0|
    +-+-+-+-+-+-+-+-+-+

     0 x x x x x x x x    Character from Screen or H. Cursor if requested
```

4.3.6 Received Cursor - Normal Communication Mode

```
    +-+-+-+-+-+-+-+-+-+
    |8|7|6|5|4|3|2|1|0|
    +-+-+-+-+-+-+-+-+-+

     1 0 0 x x x x x x    H. Cursor Position ($00-3F) Vert does not follow
     1 0 1 0 0 x x x x    H. Cursor Position ($40-4F) Vert does not follow
     1 0 1 0 1 0 0 0 0    H. Curs Pos >4F No Vert Follows - see note
     1 0 1 0 1 0 0 0 1    Reserved
     1 0 1 0 1 0 0 1 x    Reserved
     1 0 1 0 1 0 1 x x    Reserved
     1 0 1 0 1 1 x x x    Reserved
     1 0 1 1 x x x x x    Reserved
     1 1 0 x x x x x x    H. Cursor Position ($00-3F) Vert follows
     1 1 1 0 0 x x x x    H. Cursor Position ($40-4F) Vert follows
     1 1 1 0 1 0 0 0 0    H. Curs Pos >4F Vert Follows - see note
     1 1 1 0 1 0 0 0 1    Reserved
     1 1 1 0 1 0 0 1 x    Reserved
     1 1 1 0 1 0 1 x x    Reserved
     1 1 1 0 1 1 x x x    Reserved
     1 1 1 1 0 x x x x    V. Cursor Position ($00-0F)
     1 1 1 1 1 0 x x x    V. Cursor Position ($10-17)
     1 1 1 1 1 1 0 0 0    V. Cursor Position for Status Line ($18)
     1 1 1 1 1 1 0 0 1    Reserved
     1 1 1 1 1 1 0 1 x    Reserved
     1 1 1 1 1 1 1 x x    Reserved
```

Note - when "Horizontal Cursor Position Greater Than $4F" byte is received, the computer may obtain the actual value with the "Request Horizontal Cursor from XEP80" command.

Appendix A   Application Notes

1) Characters may be printed over EOLs in the right margin column without causing insert lines to occur by setting the right margin at least one more column to the right.

2) The firmware of the XEP80 uses just under 2K of the available address space - thus allowing rom replacements to incorporate additional features.

3) Debug mode allows reading of the parallel port so with the addition of a latch on the XEP80 board the port becomes bidirectional.

4) If large text screen (greater than 80 columns wide) needs to be erased use the Fill XEP80 RAM with EOL command rather than the Clear Screen char. Clear Screen only clears the current 80 column 24 line high window.

5) X Cursor > $4F is handled by the handler. The handler automatically sends the Request Horizontal Cursor command to the XEP80 whenever a $50 cursor X value is returned.

6) When using the XEP80 Internal Character Set issue the Fill XEP80 RAM with Space command to blank the screen. However, when sending characters to the XEP80 screen, either precede each one with an ESC character or set the LIST flag so that screen edits do not occur.

7) Graphics 8 screen dumps will require a short discussion. Since the OS requires 8K RAM to set up a mode 8 screen the handler cannot reside in its standard address space. An appended AUTORUN.SYS file must first set the MEMTOP pointer to 8K below the top of RAM (check actual mode 8 value). Then the relocater will move the handler below this point. After entering 40 col mode the GRAPHICS 8 call may be made. Plotting etc. to the screen is through the normal S: interface of the OS. When done the application will set XEP80 mode then set XEP80 graphics, then read from the ANTIC screen RAM and output to the XEP80.

8) Changing default printer configuration at boot time is shown by disk program PRINTER.BAS on the system diskette.

Appendix B   Relocater Format for XEP80 Handler

Values in Boxes are in HEX Format

```
AUTORUN.SYS File Format            Relocating Format of Handler

+-------+                          +----+
| FF,FF | DOS Load File         +--| XX | Link byte to first
+-------+                       |  +----+   Relocate byte
| 00,30 | Start Address=$3000   |  :    :
+-------+                       |  :    : Valid 6502 Code
| LO,HI | End Address           |  +----+
+-------+                       +->| XX | High Byte to relocate
:       : Relocater                +----+   add offset
:       :   Code                +--| XX | Link byte to second
+-------+                       |  +----+   Relocate byte
| LO,HI | Handler Length        |  :    :
+-------+                       |  :    : Valid 6502 Code
|  20   | JSR Instruction       |  +----+
+-------+ for DOSINI set        +->| XX | High Byte to relocate
| LO,HI | by Relocater             +----+   add offset
+-------+                       +--| 00 | Example Link of 0
|  4C   | JMP Instruction       |  +----+   following byte is
+-------+ to INIT code          +->| XX | High byte to relocate
| LO,HI | set by Relocater         +----+
+-------+                       +--| 80 | Special Case 80 means
| 00,00 | Printer Aux bytes     |  +----+   skip 127 data bytes
+-------+                       |  :    :   then treat byte 128
:       : XEP 80 Handler        |  :    :   as the next link
:       :   Code in             +->+----+
:       : Relocating Format     +--| XX | Link
+-------+                       |  +----+
| E2,02 | INIT Vector           |  |    | Valid 6502 Code
+-------+                       |  +----+
| E3,02 | for DOS               +->| 00 | Dummy byte to Relocate
+-------+                          +----+   to maintain links
| 00,30 | RUN Address              | FF | Special Case FF means
+-------+                          +----+   end of File
```

Appendix C   Contents of System Diskette

DOS.SYS	DOS 2.5
DUP.SYS	
AUTORUN.SYS	XEP80 Relocater and Handler
XEP80.DOC	This Document
XEP80HAN.SRC	Handler in assembly source
RELOC.SRC	Relocater in assembly source
MAKER.BAS	Program to generate AUTORUN.SYS
WINDOW.BAS	Examples for Owners Manual
EIGHTY.BAS	
PRINTER.BAS	
GRAPHICS.BAS	
DEMO80.BAS	Demonstration program for XEP80